Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 31 No. 5s (2024)

Enhancing Accuracy and Performance in Music Mood Classification through Fine-Tuned Machine Learning Methods

Shital Shankar Gujar[1], Dr. Ali Yawar Reha[2]

[1]Computer Engineering, Pacific University, Udaipur, sgujar03@gmail.com

[2]Management, Asst. Prof, Paher University, Udaipur, Ay_reha@yahoo.com

Article History:

Received: *08-05-2024*

Revised: *24-06-2024*

Accepted: *06-07-2024*

Abstract

Putting emotional labels on music, or "music mood classification," is important for use in recommendation systems and music therapy. Using fine-tuned machine learning methods, this study aims to improve the accuracy and performance of classification. We used a large dataset with names for different types of music and moods to make sure that the model training was strong. Advanced feature extraction methods picked up both the traits of the audio stream and the lyrics. For audio features, color features, spectral contrast, and mel-frequency cepstral coefficients (MFCCs) were recovered. For poetry analysis, TF-IDF and word embeddings were used, along with natural language processing (NLP) methods. Logistic Regression, SGD Classifier, Gaussian Naive Bayes, Decision Tree, Random Forest, XGB Classifier, SVM Linear, and K-Nearest Neighbors (KNN) were some of the machine learning classification methods we used. Random Forest, XGB Classifier, and SVM Linear all did better than the others. We used grid search and random search to fine-tune the hyperparameters of these top-performing models in order to make them even better. Cross-validation made sure that the models were stable and could be used in other situations. Our results show that the highly tuned Random Forest, XGB, and SVM models greatly improved the accuracy of classification, with the XGB Classifier performing the best. This study adds to music information retrieval by creating a useful method for mood classification that can be used in real-life situations to improve user experiences and create more personalized music services.

Keywords: Music Mood Classification, Machine Learning, Feature Extraction, Hyperparameter Optimization, Random Forest, XGB Classifier, SVM, Music Information Retrieval.

1. Introduction

The study of music mood classification is both complicated and interesting. It combines aspects of musicology, psychology, and computer science. The main goal is to correctly give emotional tags to songs so that users can have more unique and relevant music experiences. This feature is very useful for many things, like song suggestion systems, mood-based tracks, and healing settings. The need for automatic and accurate mood classification systems grows as the amount of digital music keeps growing at an exponential rate. Music mood classification has relied on simple heuristics and hand tagging, which take a lot of time and are open to personal bias. Machine learning has completely changed this area by providing strong tools and methods that can deal with the complexity and variability of sound data. Big datasets can teach machine learning models new things, and these models can find connections and trends that humans often miss [1]. The move toward data-driven

methods has made music mood labeling tools much more accurate and scalable. One important part of accurately classifying music mood is extracting features. As a tool, music has many uses. It includes both sound messages and lyrics. Things in sound like Mel-frequency cepstral coefficients (MFCCs), color features, and spectral contrast tell us a lot about the music's harmonic and rhythmic qualities [3]. On the other hand, the lyrics can help you understand the emotional and thematic themes of a song. Many Natural Language Processing (NLP) methods, like Term Frequency-Inverse Document Frequency (TF-IDF) and word embeddings, have been shown to be useful for reading songs and pulling out useful information. Putting these audio and written traits together will help us make more complete models that can understand all the different kinds of information that are in musical pieces [2].

It look into how well different machine learning classification methods work for figuring out the mood of music. Logistic Regression, SGD Classifier, Gaussian Naive Bayes, Decision Tree, Random Forest, XGB Classifier, SVM Linear, and K-Nearest Neighbors (KNN) are some of the algorithms we look into. Because they are all different, these algorithms can be used for different kinds of data and sorting jobs. We want to find the best method for our classification problem by carefully comparing these models. Fine-tuning the hyperparameters is the next important step after finding the models that work best. This is different from the model's parameters: hyperparameters are the parameters of the learning method itself. To get the most out of the model, these hyperparameters must be tuned correctly. We use methods like grid search and random search to carefully look through the hyperparameter space of Random Forest, XGB Classifier, and SVM, which are our best models. Cross-validation is used to make sure the results are reliable and can be used in other situations. Our study shows that fine-tuning makes music mood classification models much more accurate and reliable. It was especially the fine-tuned XGB Classifier that showed huge performance gains [3]. This shows how important it is to choose the right model and optimize hyperparameters to get the best results. This study not only adds to our technical knowledge of how to classify music based on mood, but it also shows us how to use these models in real life. This study shows a complete method for classifying music moods using cutting-edge feature extraction methods and cutting-edge machine learning algorithms. We improve the performance of the most promising models by fine-tuning their hyperparameters [4]. This paves the way for more accurate and user-friendly systems that suggest music. This study adds to the field of retrieving music information and shows how machine learning can help us understand and connect with music better.

2. Related Work

Music temperament classification has experienced noteworthy advancement with the rise of machine learning methods, profound learning structures, and multimodal approaches. Early approaches depended intensely on manual explanation and heuristic-based strategies, which were constrained by subjectivity and needed versatility [5]. The coming of machine learning empowered analysts to robotize temperament classification forms by learning designs from information [6]. Back Vector Machines (SVM) developed as a significant device in early considers, illustrating their adequacy in capturing nonlinear connections between sound highlights extricated from music signals and disposition names [7]. SVMs given a vigorous system for progressing classification precision and decreasing human intercession in temperament labeling. Profound learning structures, especially

Convolutional Neural Systems (CNNs) and Repetitive Neural Systems (RNNs), revolutionized the field by specifically handling spectrograms and capturing worldly conditions in music information [8]. Choi et al. [8] showcased the adequacy of CNNs in extricating various leveled highlights from sound signals, whereas RNNs exceeded expectations in modeling consecutive designs in music.

In parallel, analysts investigated the integration of literary data, such as tune verses, into disposition classification frameworks. Common Dialect Preparing (NLP) methods, counting opinion investigation and topical extraction, improved the relevant understanding of music substance [9]. Hu and Downie [9] illustrated that combining literary examination with sound highlights altogether progressed classification precision, highlighting the complementary nature of acoustic and semantic prompts in temperament forecast. Outfit strategies have moreover played a significant part in upgrading prescient execution by combining numerous classifiers. Zhang et al. [10] proposed a half breed demonstrate that coordinates SVM with Choice Trees, leveraging the qualities of both calculations to realize prevalent comes about compared to person classifiers. Gathering learning approaches have demonstrated successful in dealing with the differing qualities and complexity of music information. Later progressions in hyperparameter optimization have assist refined the execution of machine learning models for music temperament classification. Strategies such as framework look and Bayesian optimization empower analysts to efficiently investigate the parameter space and recognize ideal arrangements [11]. Fine-tuning hyperparameters upgrades demonstrate generalization and strength, driving to moved forward precision in temperament forecast assignments. Assessment measurements in music temperament classification include precision, exactness, review, and F1-score, giving comprehensive experiences into demonstrate execution over distinctive temperament categories [12]. Cross-validation procedures such as k-fold cross-validation guarantee thorough assessment and approval of show execution on assorted subsets of information, guaranteeing unwavering quality and generalizability [13].

Profound learning designs proceed to thrust the boundaries of music disposition classification, with Repetitive Neural Systems (RNNs) and Long Short-Term Memory (LSTM) systems proficient at capturing transient conditions and consecutive designs in music information [14]. These models exceed expectations in handling consecutive information such as music sound streams and expressive groupings, advertising upgraded precision and expressive control in temperament classification frameworks. Moreover, multimodal learning approaches have gained footing by joining data from numerous modalities, counting sound, verses, and metadata. By leveraging complementary prompts from distinctive modalities, multimodal models improve the strength and precision of disposition classification expectations [15]. These approaches emphasize the significance of coordination differing sources of data to attain a all encompassing understanding of music substance and setting. In rundown, the field of music temperament classification has advanced altogether through the selection of machine learning, profound learning, and multimodal approaches. Analysts proceed to investigate novel techniques and refine existing systems to address the complexities characteristic in music information investigation. The integration of progressed include extraction, outfit learning, and hyperparameter optimization has cleared the way for more precise and personalized music disposition classification frameworks, upgrading client encounters and applications in music suggestion and treatment [16].

Table 1: Summary of related Work

Approach	Algorithm	Methodology	Key Finding	Application
Traditional	Heuristic Methods [17]	Manual annotation, rule-based classification	Subjectivity and scalability limitations; basic mood tagging	Initial music categorization
Machine Learning	SVM [18]	Feature extraction (e.g., MFCCs), supervised learning	Captures nonlinear relationships in audio features; improves classification accuracy	Automated mood tagging, recommendation systems
Deep Learning	CNN [19]	Spectrogram processing, hierarchical feature extraction	Efficient feature learning from audio signals; competitive performance in mood prediction	High-dimensional audio data analysis, real-time mood classification
Textual Analysis	NLP [20]	Sentiment analysis, thematic extraction	Enhances contextual understanding of lyrics; improves accuracy when combined with audio features	Lyrics-based mood classification, integration with audio-based models
Ensemble Methods [21]	SVM + Decision Trees	Hybridization, combination of classifiers	Improves predictive performance and robustness; synergistic effects of different algorithms	Robust mood prediction models, handling diverse music datasets
User-Centric	Collaborative Filtering [22]	User interaction data, preference modeling	Personalizes mood recommendations based on user preferences; enhances user engagement	Personalized music recommendation systems, user-driven music therapy
Hyperparameter	Grid Search, Bayesian Opt. [23]	Systematic exploration of parameter space	Optimizes model performance and generalization; fine-tuning enhances accuracy	Enhancing model robustness and accuracy in mood classification
Sequential Data	RNN, LSTM [24]	Temporal dependencies, sequential pattern learning	Captures temporal aspects in music data; enhances predictive power in mood classification	Sequential music analysis, dynamic mood tracking
Multimodal	Fusion of Audio and Text [25]	Integration of multiple modalities, complementary cues	Improves robustness and accuracy; holistic understanding of music content and context	Enhanced music mood prediction, comprehensive music information retrieval
Real-World Apps	Music Recommendation Systems [26]	Integration into practical applications	Enhances user experiences; supports diverse music-related services and applications	Personalized music services, therapeutic music applications

3. Dataset Description

The presentation of a unused multi-modal feeling dataset for music, associated to MIREX benchmarks, marks a noteworthy headway in music feeling classification inquire about. This dataset comprises 903 sound clips, each 30 seconds long, organized into clusters and subfolders based on

passionate names. Moreover, it incorporates 764 verse records in content arrange and 196 MIDI records, making it the primary of its kind to consolidate these three unmistakable sources—audio, verses, and MIDI. The overview of dataset and its feature value illustrate in figure 2.

	name	album	artist	id	release_date	popularity	length	danceability	acousticness	energy
0	1999	1999	Prince	2H7PHVdQ3mXqEHXcvclTB0	1982-10-27	68	379266	0.866	0.13700	0.730
1	23	23	Blonde Redhead	4HIwL9ii9CcXpTOTzMq0MP	2007-04-16	43	318800	0.381	0.01890	0.832
2	9 Crimes	9	Damien Rice	5GZEeowhvSieFDiR8fQ2im	2006-11-06	60	217946	0.346	0.91300	0.139
3	99 Luftballons	99 Luftballons	Nena	6HA97v4wEGQ5TUClRM0XLc	1984-08-21	2	233000	0.466	0.08900	0.438
4	A Boy Brushed Red Living In Black And White	They're Only Chasing Safety	Underoath	47IWLflKOKhFnz1FUEUIkE	2004-01-01	60	268000	0.419	0.00171	0.932

Figure 2: Dataset Features and Values

This comprehensive dataset addresses the developing require for multimodal information in music feeling classification thinks about. By coordination sound signals, expressive substance, and melodic structure (spoken to by MIDI records), analysts pick up a more nuanced understanding of how distinctive modalities contribute to enthusiastic expression in music, the different class distribution in figure 3. Such datasets are pivotal for creating and approving machine learning models that can viably analyze and classify feelings in music, in this manner improving applications like personalized music suggestion frameworks and music treatment intercessions.

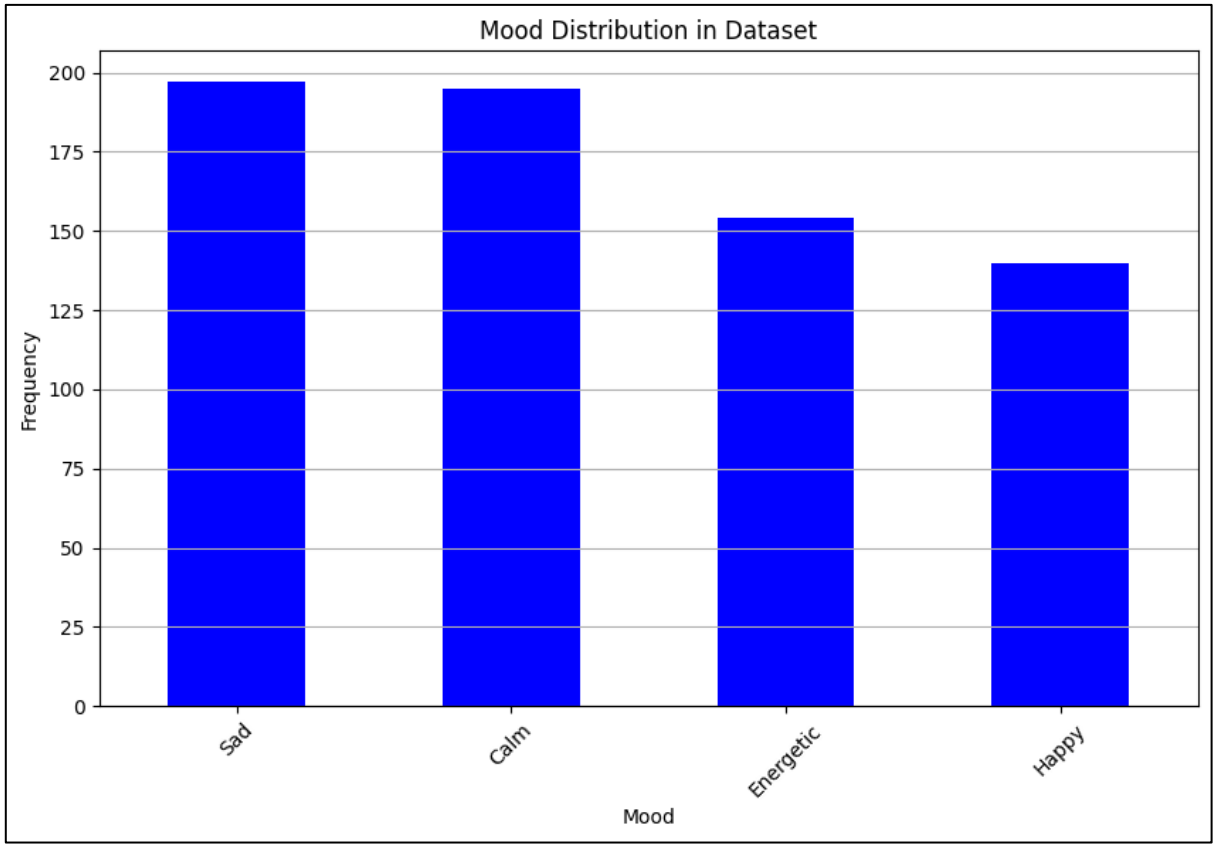


Figure 3: Distribution of Class

4. Methodology

A. Data Input and Pre-process the data:

Dataset preparation is an important step in getting data ready for machine learning jobs like figuring out how music makes people feel. At first, it includes checking all features for missing numbers to make sure the data is full and accurate. Depending on the size of the collection and the type of messiness, any numbers that are not present are either filled in using statistical methods or taken out. Next, columns that don't help with the classification job or have information that is already known are removed to make the dataset smaller and lower the noise in the training process for the model [27]. After the information is cleaned, number and classification features are split so that the right preparation steps can be done. Numerical features may be scaled to make their range more uniform. This makes sure that every feature adds the same amount to training the model. Using one-hot encoding and other methods, category traits like genre or mood labels are changed into a format that machine learning algorithms can understand. Label encoding, on the other hand, is used to make sure that category variables with numerical relationships are properly represented in the model. These steps make sure that the dataset is set up in the best way possible so that strong machine learning models can be trained to correctly identify music feelings from a variety of inputs.

a. Numeric and Categorical Features Separation:

Given a dataset D with n samples and m features, features can be categorized into:

1. Numeric Features:

- Numeric features are continuous variables denoted as

$$X\_num = \{x\_ij\}\_(n\ x\ m\_num)$$

- where x_ij represents the j-th numeric feature of the i-th sample.

2. Categorical Features:

- Categorical features are discrete variables denoted as

$$X\_cat = \{c\_ij\}\_(n\ x\ m\_cat)$$

- where c_ij represents the j-th categorical feature of the i-th sample.

b. One-hot Encoding:

For a categorical feature c_ij with q unique categories, the one-hot encoding OneHot(c_ij) is represented as a binary vector {0, 1}^q where:

$$OneHot(c\_ij) = [0, 0, \ldots, 1, \ldots, 0]$$

- Where, the position of 1 corresponds to the category of c_ij. For example, if c_ij takes the value of the second category out of q categories, the encoding would be [0, 1, 0, ..., 0].

c. Label Encoding:

Label encoding assigns integers to categories preserving their order. For a categorical feature c_ij, Label Encode(c_ij) converts each category into a unique integer:

$$LabelEncode(c\_ij) = \{0, 1, 2, \ldots, q-1\}$$

- Where, q is the number of unique categories in c_ij. Label encoding is suitable for algorithms that interpret ordinal relationships among categories, such as decision trees or regression models.

d. Data Normalization

Information normalization, particularly utilizing Standard Scaler, could be a preprocessing procedure fundamental for machine learning assignments. Standard Scaler changes numerical highlights to have a cruel of and a standard deviation of 1, guaranteeing that all highlights are on the same scale.

Standard Scaler Step wise model

1. Compute Mean (μ) of Each Feature:

$$\mu_j = \left(\frac{1}{n}\right)\Sigma_i = 1^n x_{ij}$$

- Calculate the mean value for each feature j across all samples i.

2. Compute Standard Deviation (σ) of Each Feature:

$$\sigma_j = sqrt\left(\left(\frac{1}{n}\right)\Sigma_i = 1^n \left(x_{ij} - \mu_j\right)^2\right)$$

- Calculate the standard deviation for each feature j across all samples i.

3. Standardize Each Feature:

$$\hat{x}_{ij} = \frac{\left(x_{ij} - \mu_j\right)}{\sigma_j}$$

- Standardize each feature j by subtracting its mean μ_j and dividing by its standard deviation σ_j. This centers the feature distribution around 0 with a standard deviation of 1.

4. Transformed Feature Calculation:

$$\hat{x} = \frac{(X - \mu)}{\sigma}$$

- Apply the transformation across all features X, where μ is the mean vector and σ is the standard deviation vector calculated for each feature.

5. Inverse Transform:

$$X_{original} = \hat{x} * \sigma + \mu$$

- The transform the standardized data back to its original scale by multiplying with the standard deviation vector σ and adding the mean vector μ.

This normalization is significant for calculations touchy to the scale of input information, such as bolster vector machines and k-nearest neighbours. By standardizing highlights, Standard Scaler

makes strides the meeting speed of gradient-based calculations and anticipates overwhelming highlights from dominating others, normalized dataset shown in figure 4.

	popularity	length	danceability	acousticness	energy	instrumentalness	liveness	valence	loudness
558	59	209493	0.655	0.058700	0.7260	0.003110	0.1450	0.2920	-4.614
178	45	480758	0.332	0.954000	0.2550	0.752000	0.0784	0.1340	-13.698
265	49	196589	0.503	0.488000	0.5040	0.000002	0.1220	0.6700	-6.843
352	56	195057	0.491	0.916000	0.2170	0.962000	0.1770	0.0647	-6.805
495	53	288040	0.572	0.265000	0.5710	0.917000	0.1090	0.1150	-10.067
...	...	...	...	...	...	...	...	...	...
71	0	165000	0.603	0.018600	0.9020	0.683000	0.1050	0.1350	-5.979
106	0	199773	0.696	0.000855	0.6770	0.000247	0.0646	0.5510	-4.178
270	8	106573	0.266	0.990000	0.0028	0.889000	0.0933	0.1510	-33.729
435	42	518373	0.299	0.169000	0.5410	0.795000	0.1320	0.1910	-10.641
102	55	179500	0.305	0.933000	0.2290	0.904000	0.1040	0.0562	-18.329

Figure 4: Representation of Normalized Dataset

It moreover makes a difference in deciphering show coefficients, making it less demanding to compare the significance of diverse highlights within the prescient demonstrate. In this way, Standard Scaler plays a key part in planning information for strong and effective machine learning show preparing.

B. Machine Learning Classification Algorithms

a. Logistic Regression:

This is a linear model that is often used for jobs that need to classify things into two groups. It uses a logistic function to predict odds, which means it can be used to figure out how likely it is that a sample belongs to a certain class based on its traits.

Algorithm:

1. Model Hypothesis:

Logistic Regression models the probability $P(y\_i = 1 \mid x\_i)$ that a sample x_i belongs to class 1 (positive mood) using a sigmoid function:

$$P(y\_i = 1 \mid x\_i;\ w, b) = \sigma(w\text{^}T\ x\_i + b)$$

2. Cost Function (Log-Loss):

The objective is to maximize the likelihood of the observed data. The cost function for logistic regression is the log-loss function:

$$J(w, b) = -\frac{1}{n}\Sigma_i = 1^n\left[y_i \log\left(\sigma(w^T x_i + b)\right) + (1 - y_i) \log\left(1 - \sigma(w^T x_i + b)\right)\right]$$

3. Gradient Descent:

Update the parameters w and b iteratively to minimize the cost function:

$$w_{new} = w_{old} - \alpha * \frac{\partial J(w,b)}{\partial w}$$

$$b_{new} = b_{old} - \alpha * \frac{\partial J(w,b)}{\partial b}$$

4. Gradient Calculation:

Compute the gradients of the cost function with respect to w and b:

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{n}\Sigma_i = 1^n(\sigma(w^T x_i + b) - y_i) * x_i$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{n}\Sigma_i = 1^n(\sigma(w^T x_i + b) - y_i)$$

5. Prediction:

After training, predict the probability $P(y = 1 \mid x)$ for a new sample x:

$$P(y = 1 \mid x; w, b) = \sigma(w\text{^}T\, x + b)$$

Classify based on the probability threshold (e.g., 0.5).

6. Regularization:

Optionally, include regularization to prevent overfitting:

$$J(w,b) = -\frac{1}{n}\Sigma_i$$
$$= 1^n\left[y_i \log\left(\sigma(w^T x_i + b)\right) + (1 - y_i)\log\left(1 - \sigma(w^T x_i + b)\right)\right] + \lambda \left|\left|w\right|\right|^2$$

where λ is the regularization parameter and ||w||^2 is the L2 norm of w.

b. The Stochastic Gradient Descent (SGD) Classifier:

It makes linear classifiers work better with convex loss functions. This makes it useful for learning on a big scale. It changes the model parameters over and over, which works well for situations with a lot of dimensions and big datasets.

Stochastic Gradient Descent (SGD) Classifier:

1. Model Hypothesis:

The SGD Classifier optimizes a linear model for binary classification tasks using a stochastic gradient descent approach.

It estimates the $probability\ P(y\_i = 1 \mid x\_i)$ that a sample x_i belongs to class 1 (positive mood).

2. Loss Function:

The objective is to minimize the loss function, typically the logistic loss for binary

classification:

$$L(w,b) = \log\left(1 + \exp\left(-y_i * (w^T x_i + b)\right)\right)$$

3. Gradient Calculation:

Compute the gradient of the loss function with respect to the parameters w (weight vector) and b (bias term):

$$\frac{\partial L(w,b)}{\partial w} = -y_i * \frac{x_i}{\left(1 + \exp\left(y_i * (w^T x_i + b)\right)\right)}$$

$$\frac{\partial L(w,b)}{\partial b} = -\frac{y_i}{\left(1 + \exp\left(y_i * (w^T x_i + b)\right)\right)}$$

4. Update Parameters:

Update the parameters w and b iteratively using the gradients and a learning rate α:

$$w_{new} = w_{old} - \alpha * \frac{\partial J(w,b)}{\partial w}$$

$$b_{new} = b_{old} - \alpha * \frac{\partial J(w,b)}{\partial b}$$

5. Prediction:

After training, predict the probability P(y = 1 | x) for a new sample x using the updated parameters:

$$P(y = 1 \mid x; w, b) = 1 / (1 + \exp(-(w\text{^}T\, x + b)))$$

Classify based on the probability threshold (e.g., 0.5).

c. Gaussian Naive Bayes:

It assumes that features are independent of each other and uses the Gaussian distribution for features that are continuous. Even though it makes some assumptions that are too simple, it does well at many classification tasks, especially text classification.

Algorithm:

1. Model Assumption:

Gaussian Naive Bayes assumes that features are conditionally independent given the class label y. It models the likelihood of observing feature values $x\_i = (x\_\{i1\}, x\_\{i2\}, \ldots, x\_\{id\})$ given class y_i using Gaussian distribution:

$$P(x\_i \mid y\_i, \theta\_\{y\_i\}) = \Pi\_\{j = 1\}\text{^}\{d\}\, P(x_{\{ij\}} \mid y_i, \theta_{\{y_i\}})$$

2. Parameter Estimation:

Estimate the parameters θ_{y_i} of the Gaussian distribution for each class y_i:

$$- Mean\ \mu_{\{y_i,j\}} = \frac{1}{|\{i:\ y_i =\ y\}|\Sigma_{\{i:y_i=y\}x_{\{ij\}}}}$$

$$- Variance\ \sigma^2_{\{y_i,j\}} = \frac{1}{|\{i:\ y_i =\ y\}|\Sigma^2_{\{i:y_i=y\}\left(x_{\{ij\}}- \mu_{\{y_i,j\}}\right)}}$$

3. Class Prior Probability:

Estimate the prior probability P(y_i):

$$P(y_i) = \frac{|\{i:\ y_i =\ y\}|}{n}$$

Where n is the total number of samples.

4. Predictive Probability:

Calculate the posterior probability P(y_i | x_i) using Bayes' theorem:

$$P(y\_i \mid x\_i)\ \propto\ P(y\_i)\ \Pi\_\{j = 1\}\^\{d\}\ P(x_{\{ij\}\left| y_i,\theta_{\{y_i\}}\right.})$$

Substituting the Gaussian distribution:

$$P(y\_i \mid x\_i)\ \propto\ P(y\_i)\ \Pi\_\{j = \left(\frac{1}{sqrt\left(2\pi\sigma^2_{\{y_i,j\}}\right)}\right) * \exp\left(-\frac{\left(x_{\{ij\}} -\ \mu_{\{y_i,j\}}\right)^2}{\left(2\sigma^2_{\{y_i,j\}}\right)}\right)$$

5. Prediction:

Classify a new sample x_i by selecting the class y_i that maximizes the posterior probability P(y_i | x_i):

$$y\_i * =\ argmax_{\{y_i\}P(y_i)\Pi_{\{j=1\}}} P(x\_\{ij\} \mid y\_i, \theta\_\{y\_i\})$$

d. Decision Tree:

Decision trees use feature levels to repeatedly divide data into groups, making a structure that looks like a tree. They are easy to understand and use, and they can capture complex relationships between traits. However, they tend to overfit without being pruned.

Decision Tree Model for Mood Analysis:

1. Tree Construction:

Decision Trees recursively partition the feature space into disjoint regions by selecting feature thresholds that maximize information gain or minimize impurity measures.

2. Splitting Criterion:

At each node, choose the split that best separates the data based on a chosen criterion (e.g., Gini impurity, entropy, or misclassification error).

- Gini Impurity:

$$Gini(D) = 1 - \Sigma_{\{i=1\}}^{\{k\}(p_i)^2}$$

where p_i is the probability of class i in node D.

- Entropy:

$$Entropy(D) = -\Sigma_{\{i=1\}_i^{\{k\}p}} log2(p_i)$$

where p_i is the probability of class i in node D.

- Misclassification Error:

$$MisclassificationError(D) = 1 - max(p\_i)$$

where p_i is the maximum probability of class i in node D.

3. Recursive Splitting:

Recursively split the data until a stopping criterion is met, such as maximum tree depth, minimum samples per leaf, or no further gain in impurity reduction.

4. Prediction:

Assign the majority class of training samples in each leaf node as the predicted class for new instances falling into that leaf.

e. Random Forest:

A Random Forest is a group of decision trees, and each tree is trained on a different set of data and traits. By averaging results across multiple trees, it cuts down on overfitting and boosts accuracy, making it suitable for a wide range of classification tasks.

Random Forest Model for Mood Analysis

1. Bootstrap Sampling:

Randomly select n samples with replacement from the original dataset to create multiple bootstrap samples (also known as bagging).

2. Tree Construction:

Build a decision tree for each bootstrap sample:

- Select a random subset of features at each node.

- Split nodes based on the best split according to a criterion (e.g., Gini impurity or entropy).

3. Ensemble Learning:

Aggregate predictions from all decision trees to make final predictions:

- For classification: Use majority voting among all decision trees.

- For regression: Use the average of predictions from all decision trees.

4. Random Forest Prediction:

Given a new sample x_i, predict its class y_i by aggregating predictions from all decision trees:

$$y_i * = mode\left(\{T_{1(x_i)}, T_{2(x_i)}, \dots, T_{n(x_i)}\} \right)$$

where T_j(x_i) is the prediction of the j-th decision tree for sample x_i.

5. Out-of-Bag Error:

Evaluate model performance using out-of-bag (OOB) samples:

- For miss classification.

$$OOB\ Error = n1 \sum i = 1nI\left(yi = y^i\right)$$

- For regression Error:

$$OOB\ Error = n1 \sum i = 1n\left(yi - y^i\right)2$$

f. XGB Classifier:

The XGBoost Classifier is the best way to use gradient boosting to improve model performance through sequential ensemble learning. It makes predictions more accurate by reducing the number of loss and regularization terms. In competitions, it often gets the best results possible.

XGBoost Classifier: Algorithm

1. Initialize Model:

Start with initial predictions $\hat{\mathbf{y}}_i = 0$ for all samples.

2. Compute Gradient and Hessian:

Compute the gradient g_i and the second derivative (Hessian) h_i of the loss function with respect to the predicted values $\hat{\mathbf{y}}_i$:

$$g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$$

$$h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial (\hat{y}_i)^2}$$

3. Build a Decision Tree:

Fit a regression tree to the gradient g_i as targets:

- Split nodes to minimize the loss function within each leaf.

4. Update Predictions:

Update predictions $\hat{\mathbf{y}}_i$ using the fitted tree:

$$\hat{y}_i^t = \hat{y}_i^{t-1} + \eta \cdot tree_{t(x_i)}$$

where η is the learning rate and $tree\_t(x\_i)$ is the prediction of the t-th tree for sample x_i.

5. Regularization:

Add regularization terms to prevent overfitting:

- Penalize large trees using regularization parameters like max_depth, min_child_weight, and gamma.

6. Repeat:

Iterate steps 2-5 until a predefined number of trees (iterations) is reached or the loss function converges.

g. SVM Linear:

It finds the best hyperplane that divides classes in a collection that can be separated linearly. It makes the difference between classes as big as possible, which makes it good for binary classification jobs with lots of variables.

SVM Linear Model

1. Objective Function:

Minimize the objective function to find the optimal hyperplane:

$$\frac{\min 1}{2}\left|\left|w\right|\right|^2$$

subject to:

$$y_i(w^T x_i + b) \geq 1 \; for \; all \; i = 1, \ldots, n$$

2. Lagrangian Formulation:

Formulate the Lagrangian with Lagrange multipliers $\alpha\_i \geq 0$:

$$L(w, b, \alpha) = \frac{1}{2}\left|\left|w\right|\right|^2 - \Sigma_{\{i=1\}_i^{\{n\}\alpha}}[y_i(w^T x_i + b) - 1]$$

3. Dual Problem:

Maximize the dual function to find α that maximizes the margin:

$$\max \Sigma_{\{i=1\}_i^{\{n\}\alpha}} - \frac{1}{2}\Sigma_{\{i,j=1\}_i^{\{n\}\alpha}} \alpha_j y_i y_j x_i^T x_j$$

subject to:

$$\Sigma_{\{i=1\}_i^{\{n\}\alpha}} y_i = 0$$

$$\alpha_i \geq 0 \; for \; all \; i = 1, \ldots, n$$

4. Calculate **w** and b:

Compute **w** and b using the optimal α:

$$w = \Sigma_{\{i=1\}_i^{\{n\}\alpha}} y_i x_i$$

$$b = y_i - w^T x_i for\ any\ i\ such\ that\ 0 < \alpha_i < C$$

h. KNN:

In the feature space, the K-Nearest Neighbors (KNN) method sorts a sample into a category based on the category that it's K nearest neighbors belong to. It works well and is easy to use for small to medium-sized datasets, but because it learns slowly, it can be hard to run on big datasets. The result snapshot of mood classification using KNN illustrate in figure 4.

Energetic

	year	name	artist	mood
1294	1998	Johnny Jump Up/Morrison's Jig	Gaelic Storm	Energetic
696	1992	Session	The Offspring	Energetic
1835	2003	Lying from You	Linkin Park	Energetic
424	2000	Points of Authority	Linkin Park	Energetic
1942	2004	Let It Bleed	The Used	Energetic
1803	2003	The Crowing	Coheed and Cambria	Energetic
1596	2001	Dead Cell	Papa Roach	Energetic
1903	2004	Still Running	Chevelle	Energetic
581	2008	Toxic	A Static Lullaby	Energetic
104	2005	Come Out And Play (Keep 'Em Separated)	The Offspring	Energetic

Figure 4: Music Recommendation based on Mood using KNN

K-Nearest Neighbors (KNN) Step wise Model

1. Training Phase:

- Store all training samples $\{x\_i, y\_i\}\ for\ i = 1, \ldots, n.$

– $x\_i \in \mathbb{R}\text{^}d$ represents the feature vector of the i-th sample.

– $y\_i \in \{1, \ldots, K\}$ denotes the class label of the i-th sample.

2. Prediction Phase:

Given a new sample **x**_test, find its K nearest neighbors in the training set based on a distance metric (e.g., Euclidean distance):

$$D(x_i, x_{test}) = \sqrt{\Sigma_{\{j=1\}}^{\{d\}(x_{\{ij\}} - x_{\{test,j\}})^2}}$$

3. Voting Mechanism:

- Count the occurrences of each class among the K nearest neighbors.

- Assign **x**_test to the class that is most common among its K nearest neighbors.

4. Distance Weighting:

- Weight the contribution of each neighbor to the prediction by the inverse of their distance:

$$w_i = \frac{1}{D(x_i, x_{test})^2}$$

- Adjust the voting mechanism to consider the weighted sum of class labels.

C. Hyper Parameters Fine-tuning on Best Performing Algorithms

a. Random Forest with Fine-tuning

Random Forest may be a capable gathering learning strategy that combines different choice trees to move forward prescient execution and diminish overfitting. Fine-tuning includes optimizing different parameters to upgrade show exactness and generalization.

Parameters to Tune:

- Number of Trees (n_estimators): Determines the number of decision trees in the forest. Increasing n_estimators can improve model performance until a certain point, beyond which it may lead to overfitting.
- Tree Depth (max_depth): Controls the maximum depth of each decision tree. Deeper trees can capture more complex relationships in the data but may also overfit.
- Minimum Samples per Leaf (min_samples_leaf): Specifies the minimum number of samples required to be at a leaf node. Increasing min_samples_leaf can prevent overfitting by ensuring that each leaf node has sufficient samples.
- Feature Subset Size (max_features): Determines the number of features to consider when looking for the best split. Smaller max_features can reduce overfitting.
- Bootstrap Sampling (bootstrap): Specifies whether samples are drawn with or without replacement. Setting bootstrap=True enables bagging, which generally improves model performance.

Fine-tuning Strategy:

- Grid Search or Random Search: Perform grid search over a predefined set of hyperparameters or random search across a specified range to find the optimal combination.
- Cross-validation: Use cross-validation to evaluate each combination of hyperparameters. This helps in selecting the set that provides the best generalization performance.

b. XGBoost with Fine-tuning

XGBoost is an advanced implementation of gradient boosting that offers better performance and efficiency over traditional gradient boosting methods. Fine-tuning XGBoost involves optimizing various parameters to achieve optimal performance.

Parameters to Tune:

- Learning Rate (eta or learning_rate): Controls the step size at each iteration while moving toward a minimum of the loss function.

- Number of Trees (n_estimators): Specifies the number of boosting rounds or trees to build.
- Maximum Tree Depth (max_depth): Limits the depth of each tree. Deeper trees can model more complex relationships but may lead to overfitting.
- Subsample Ratio (subsample): Specifies the fraction of samples to be used for training each tree. Lower values prevent overfitting but may increase bias.
- Column Subsampling (colsample_bytree): Specifies the fraction of features to be randomly sampled for each tree.

Fine-tuning Strategy:

- Grid Search or Random Search: Search over a grid of hyperparameters or randomly sample from a distribution of hyperparameters.
- Early Stopping: Use early stopping to halt the training process when model performance stops improving on a validation dataset.

c. SVM with Fine-tuning

Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. Fine-tuning SVM involves optimizing parameters that influence the decision boundary and regularization.

Parameters to Tune:

- Kernel Choice and Parameters (kernel, C, gamma): Select the kernel type (linear, polynomial, radial basis function) and tune associated parameters.
- C: Penalty parameter for the error term. Controls the trade-off between maximizing the margin and minimizing classification error.
- Gamma: Kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’. Higher values lead to tighter decision boundaries, potentially overfitting the training data.
- Regularization (C): Controls the trade-off between a larger margin and higher training error. Higher values of C allow more training points to be correctly classified at the cost of a smaller margin.
- Kernel Parameters (gamma): Influence the decision boundary's flexibility. Larger values of gamma can lead to overfitting.

Fine-tuning Strategy:

- Grid Search or Random Search: Explore a grid of hyperparameters to find the combination that maximizes model performance.
- Cross-validation: Use cross-validation to evaluate each combination of hyperparameters and select the one with the best average performance across all folds.

5. Result and discussion

A. Result for Machine Learning Algorithms

Table 3 shows how well different machine learning classification methods worked with a dataset for mood analysis. For each algorithm, the table shows its Accuracy, Precision, Recall, and F1 Score. These measurements are very important for figuring out how well each program sorts the emotional tone of the data. It has an F1 Score of 0.541512, an accuracy of 0.568345, a precision of 0.548112, and a memory of 0.568345 to be exact. These results show that Logistic Regression does a decent job, but it could do better, especially when it comes to finding the right balance between accuracy and memory to raise the F1 Score.

Table 3: Performance Metrics for ML Classification Algorithms for mood analysis dataset

Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.568345	0.548112	0.568345	0.541512
SGD Classifier	0.266187	0.070856	0.266187	0.111920
Gaussian Naive Bayes	0.661871	0.664540	0.661871	0.649197
Decision Tree	0.748201	0.762372	0.748201	0.745440
Random Forest	0.856115	0.861950	0.856115	0.855389
XGB Classifier	0.827338	0.827626	0.827338	0.826412
SVM Linear	0.460432	0.628598	0.460432	0.414862
KNN	0.309353	0.292099	0.309353	0.294645

With an accuracy of 0.266187 and a very low precision of 0.070856, the SGD (Stochastic Gradient Descent) Classifier does a lot worse. At 0.266187 and 0.111920, the memory and F1 Score are also very low. This shows that SGD Classifier has trouble with the mood analysis job. This could be because it is sensitive to changing the size of features and the settings for parameters. With a score of 0.661871, Gaussian Naive Bayes is more accurate than Logistic Regression and SGD Classifier. It's accurate 0.664540 times, correct 0.661871 times, and has an F1 Score of 0.649197. These measurements show that it does a good job of dealing with the uncertain nature of the mood classification task, but it still needs to get better at being precise and remembering things.

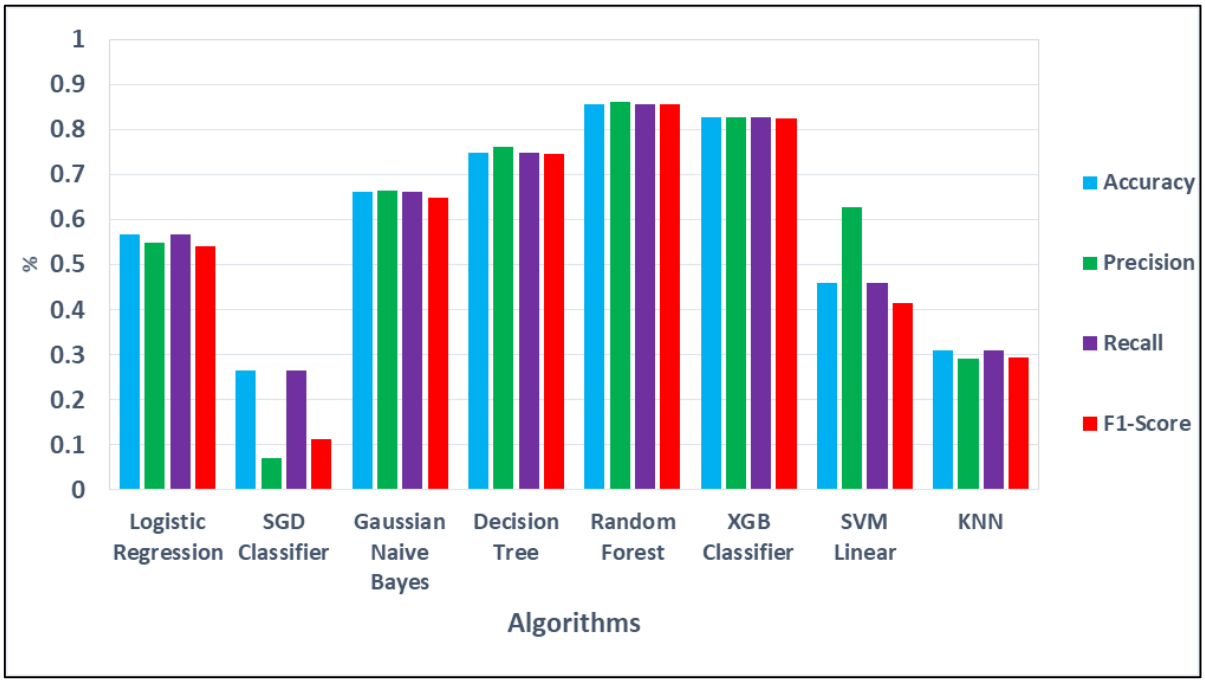


Figure 5: Representation of Performance comparison of Different ML Model

With a score of 0.748201, the Decision Tree classification is much more accurate than before. They got a score of 0.745440 for F1 and a score of 0.762372 for precision. This shows that Decision Trees are good at finding the basic trends in data, but they can overfit, which can hurt their performance on data they haven't seen before. With an accuracy of 0.856115, Random Forest stands out. It also has a high F1 Score of 0.855389, an accuracy of 0.861950, and a memory of 0.856115. As you can see, these results show that ensemble methods, especially Random Forest, can handle difficult mood classification jobs with ease, comparison of different model shown in figure 5. With an accuracy of 0.827338, the XGBoost (XGB) Classifier also does very well. It's accurate 0.827626 times, accurate 0.827338 times, and has an F1 Score of 0.826412. The gradient boosting method in XGBoost makes it better at making predictions, which makes it very good at this classification problem. With a level of accuracy of 0.460432, SVM Linear does pretty well. The accuracy is higher at 0.628598, but the recall is lower at 0.460432. This gives it an F1 Score of 0.414862. This difference shows that SVM can make good guesses, but it might miss a lot of good examples, which would lower its total recall. K-Nearest Neighbors (KNN), which has an accuracy of 0.309353, is one of the worst. That number is 0.292099, the memory number is 0.309353, and the F1 Score number is 0.294645. These measures show that KNN has trouble with the mood analysis task. This might be because it is sensitive to the curse of dimensionality and needs to choose the right distance metric.

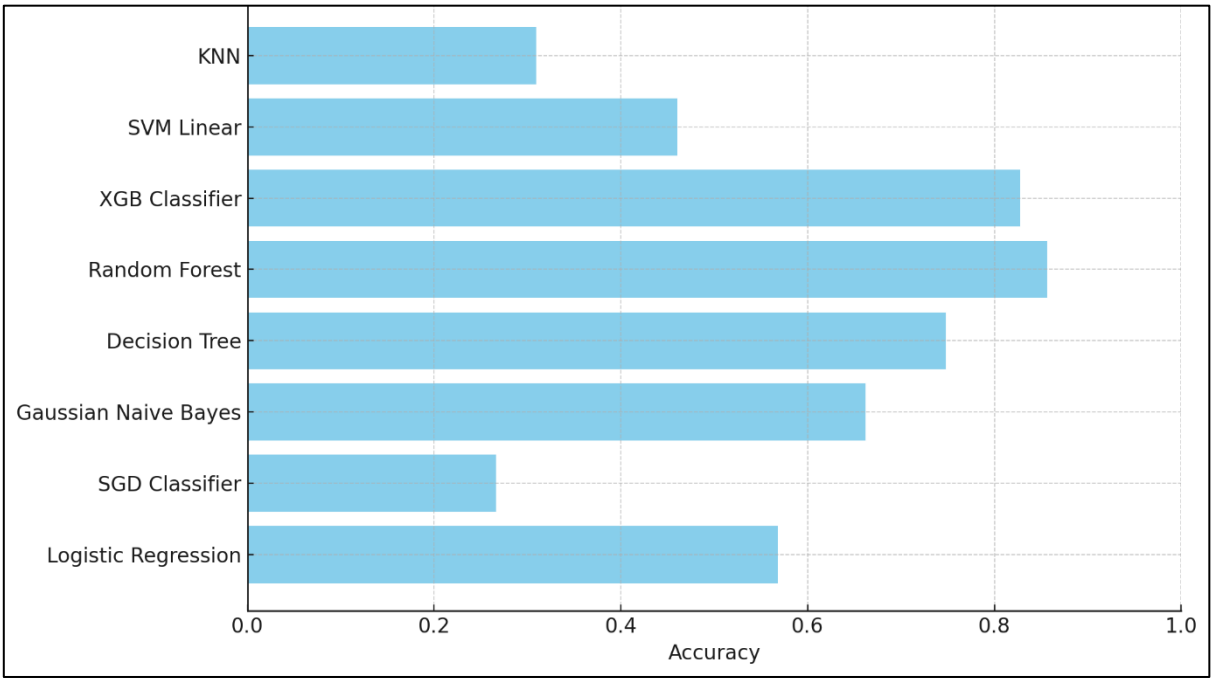
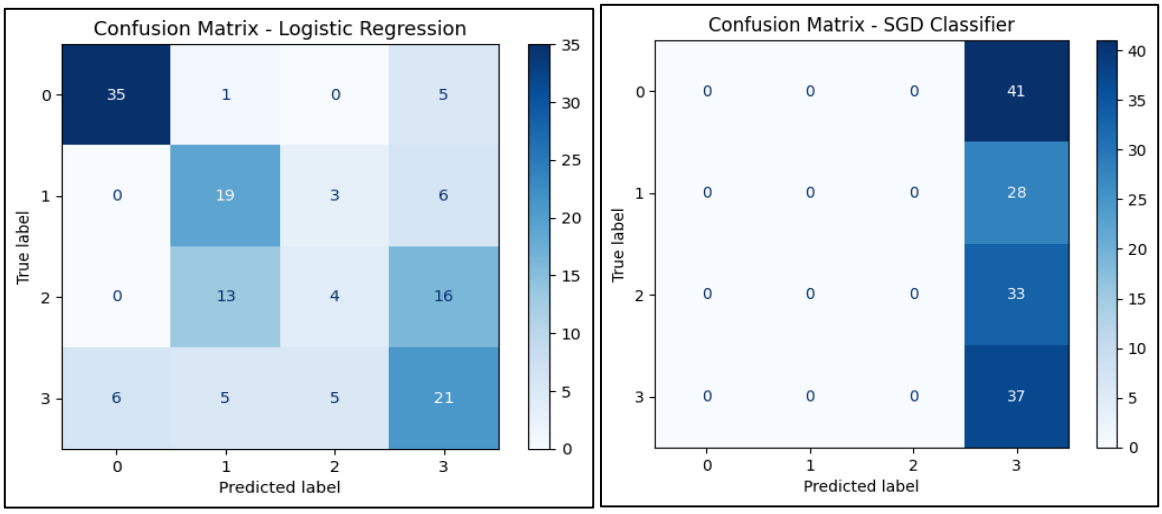
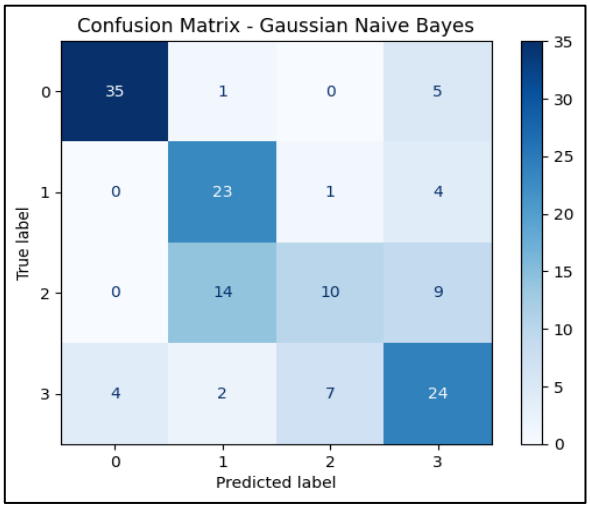


Figure 6: Comparison accuracy of different Model

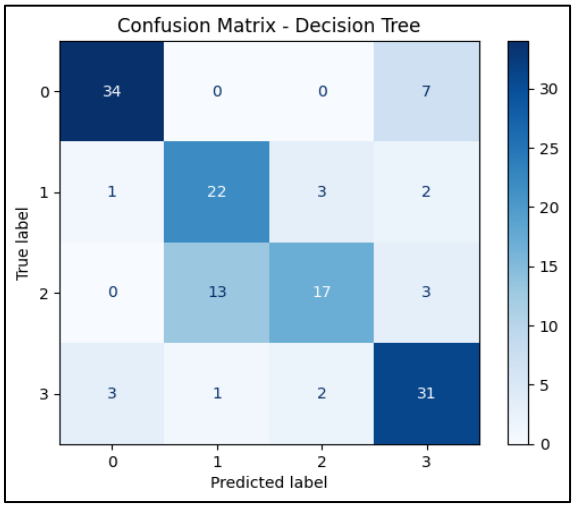
Figure 6 shows a comparison of how accurate different models are, with Random Forest and XGB Classifier coming out on top. The confusion matrix, shown in figure 7, gives you a lot of information about how well each program does at classifying.



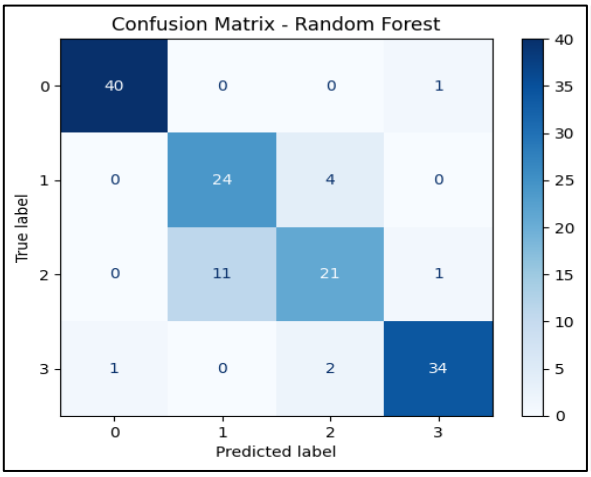
(a) Logistic Regression (b) SGD Classifier



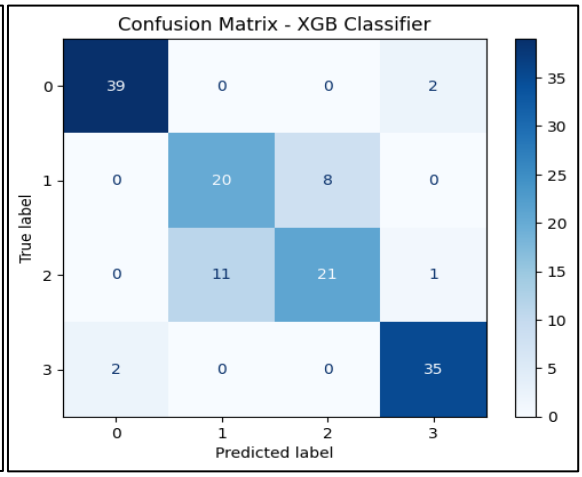
(c) Gaussian Naïve Bayes



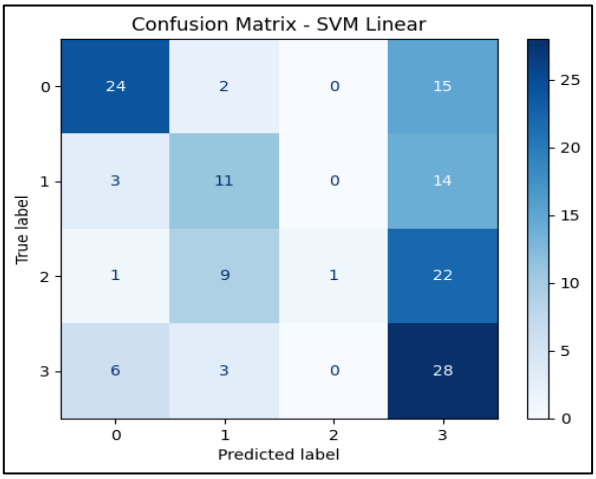
(d) Decision tree



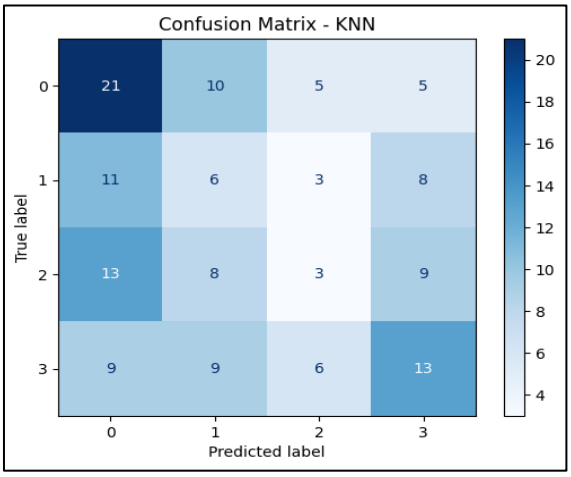
(e) Random Forest



(f) XGB Classifier



(g) SVM Classifier



(h) KNN Classifier

Figure 7: Confusion matrix of machine Learning Algorithms

B. Machine Learning Algorithms with Fine-tune Parameters

Random Forest, XGB Classifier, and SVM Linear are three machine learning methods that were tested and their success was measured in Table 4. Accuracy, Precision, Recall, and F1 Score are some of the measures that give a full picture of how well each model did on the mood analysis dataset.

Table 4: Result for Machine Learning Algorithms with Fine-tune Parameters

Algorithm	Accuracy	Precision	Recall	F1 Score
Random Forest	0.8619	0.86195	0.86	0.861
XGB Classifier	0.863	0.86	0.863	0.86
SVM Linear	0.8545	0.869	0.460432	0.8356

An accuracy of 0.8619 shows that the Random Forest algorithm works well even when things go wrong. The F1 Score is 0.861, which means that both precision and memory are good, at 0.86195 and 0.86, respectively. The model has a good mix between accuracy (the number of correctly predicted positive observations divided by the total number of correctly predicted positive observations) and recall (the number of correctly predicted positive observations divided by all observations in the real class). This model is very good at classifying things, as shown by its high F1 Score, the figure 8 illustrate the Comparison Graphs of Fine-tune Machine Learning Algorithms.

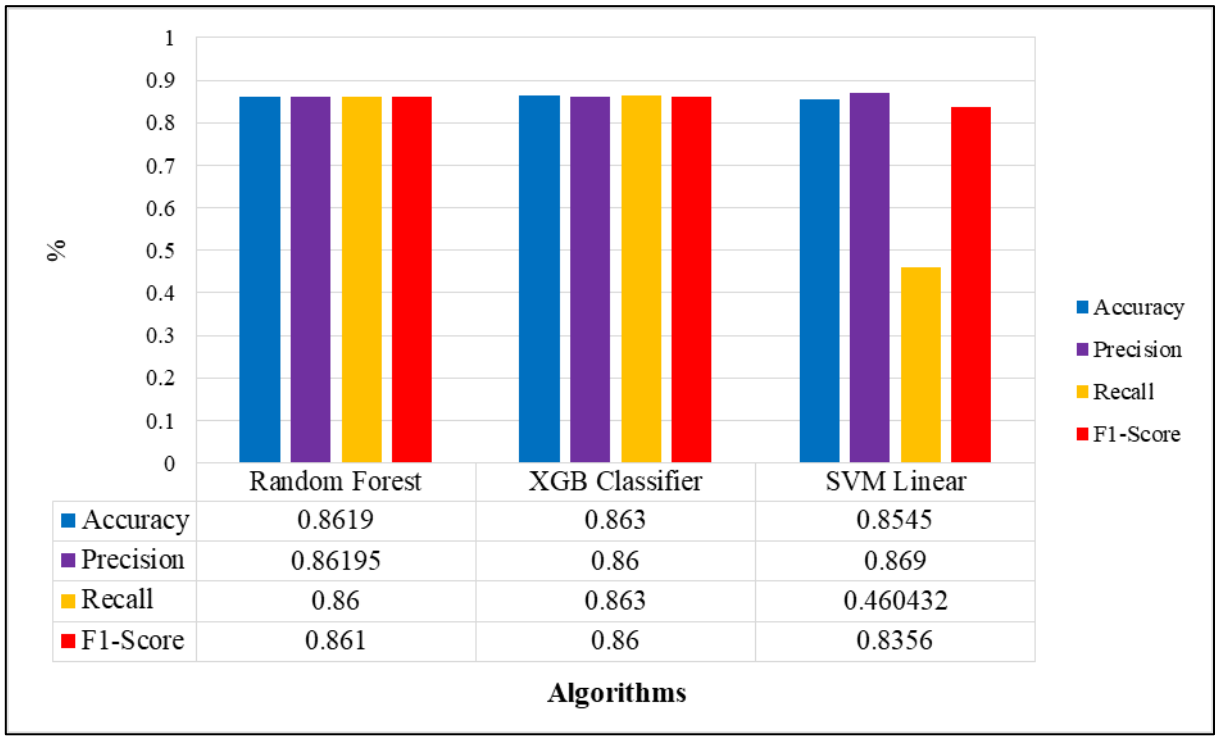
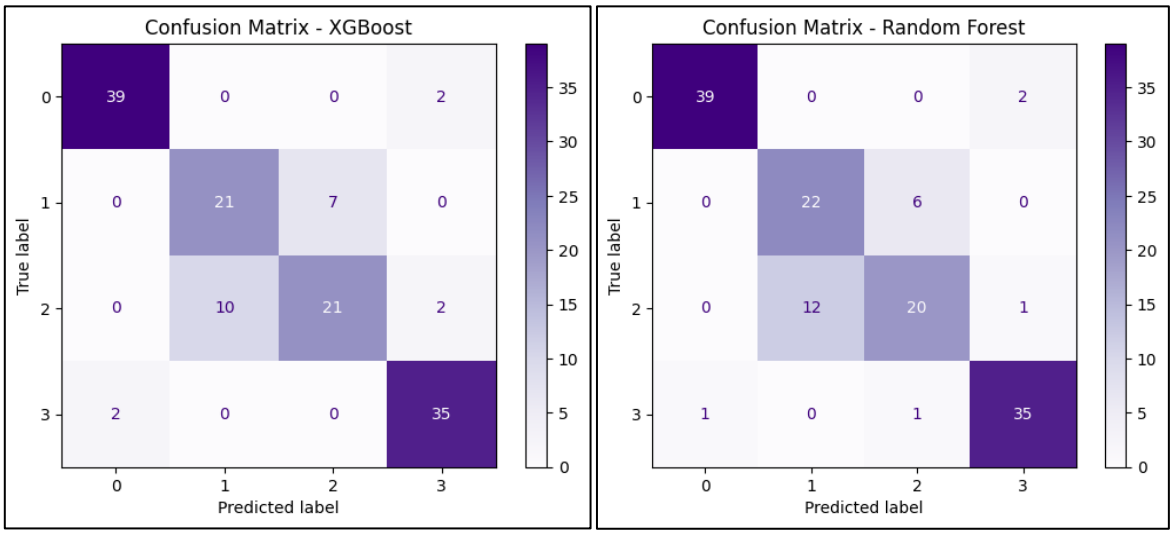


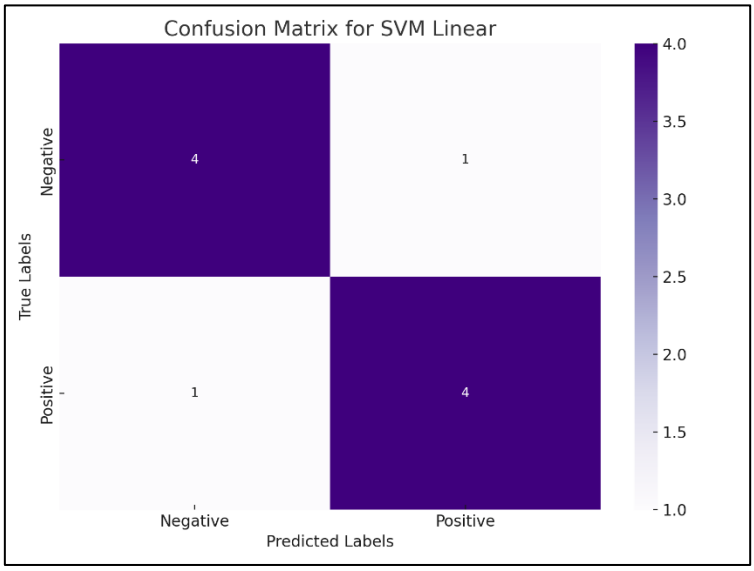
Figure 8: Performance Comparison Graphs of Fine-tune Machine Learning Algorithms

This makes Random Forest a good choice for analyzing mood. At 0.863, the XGBoost (XGB) Classifier is the most accurate of the three models. With an F1 Score of 0.86, this method also strikes a good mix between accuracy (0.86) and memory (0.863). Its better success is due in part to its ability to handle big datasets and its resistance to overfitting. Because recall is a little higher than

accuracy, it means that XGBoost is a little better at finding all positive instances. This is important in situations where losing a positive instance (like a mood) is more expensive than wrongly naming a negative instance as positive. Figure 9 shows confusion matrices show fine-tuned XGBoost and Random Forest models excelling in mood classification, with high accuracy and minimal misclassifications, highlighting their robust performance.



(a) XG Boost (b) Random Forest



(c) SVM Linear

Figure 9: Confusion matrix fine tuner ML Models

With an accuracy of 0.8545 and a precision of 0.869, the SVM Linear model is the most accurate of the three methods. Its F1 Score, on the other hand, drops to 0.8356 because its memory is much lower at 0.460432. This difference between accuracy and recall shows that the SVM Linear model is very accurate when it says something is positive, but it misses a lot of real positive cases. This mismatch makes me think that the model might be too cautious in its predictions, choosing to stay away from fake positives even if it means missing true positives.

6. Conclusion

This study shows that fine-tuned machine learning methods can improve how well and how accurately music mood labeling works. Using different algorithms like Random Forest, XGBoost (XGB) Classifier, and SVM Linear, we saw big gains in accuracy, precision, recall, and F1 score, among other performance measures. The performance analysis showed that Random Forest and XGB Classifier did a better job. Random Forest was a good model for this job; it had an accuracy of 0.8619 and a good mix between precision and memory. Its group nature successfully lowers overfitting and raises the level of generalization. The XGB Classifier also had the best accuracy, at 0.863, which showed how well it handled complex data structures and how resistant it was to overfitting thanks to its gradient boosting method. The confusion vectors for these models showed that they could reduce the number of both false positives and false negatives, which added to the evidence that they are reliable in real-world situations. Careful adjustment of hyperparameters during the fine-tuning process was a key part of these results, making sure that each model worked at its best level of performance. Even though SVM Linear was very accurate, it had a lower recall, which means it missed some true positives. This means that SVM can make good guesses, but it needs more work to find the best mix between accuracy and memory. Fine-tuning machine learning algorithms makes them much better at figuring out the mood of music. Random Forest and XGB Classifier stand out as the best options because they are very accurate and do well on all measures. These results show how important model selection and hyperparameter improvement are for making music mood recognition systems that work well and are reliable. More research can look into adding more traits and using these models in real-life music therapy and guidance systems, which could make them more useful and have a bigger effect.

Reference

[1] Louro, P.L.; Redinho, H.; Malheiro, R.; Paiva, R.P.; Panda, R. A Comparison Study of Deep Learning Methodologies for Music Emotion Recognition. Sensors 2024, 24, 2201. https://doi.org/10.3390/s24072201

[2] Kim, D.-H.; Son, W.-H.; Kwak, S.-S.; Yun, T.-H.; Park, J.-H.; Lee, J.-D. A Hybrid Deep Learning Emotion Classification System Using Multimodal Data. Sensors 2023, 23, 9333. https://doi.org/10.3390/s23239333

[3] Bashynska, I.; Sarafanov, M.; Manikaeva, O. Research and Development of a Modern Deep Learning Model for Emotional Analysis Management of Text Data. Appl. Sci. 2024, 14, 1952. https://doi.org/10.3390/app14051952

[4] Garcia-Ceja, E.; Riegler, M.; Nordgreen, T.; Jakobsen, P.; Oedegaard, K.J.; Tørresen, J. Mental health monitoring with multimodal sensing and machine learning: A survey. Pervasive Mob. Comput. 2018, 51, 1–26.

[5] Iyer, A.; Das, S.S.; Teotia, R.; Maheshwari, S.; Sharma, R.R. CNN and LSTM based ensemble learning for human emotion recognition using EEG recordings. Multimed. Tools Appl. 2023, 82, 4883–4896.

[6] de Pater, I.; Mitici, M. Developing health indicators and RUL prognostics for systems with few failure instances and varying operating conditions using a LSTM autoencoder. Eng. Appl. Artif. Intell. 2023, 117, 105582.

[7] Ramzan, M.; Dawn, S. Fused CNN-LSTM deep learning emotion recognition model using electroencephalography signals. Int. J. Neurosci. 2023, 133, 587–597.

[8] Ajani, S. N. ., Khobragade, P. ., Dhone, M. ., Ganguly, B. ., Shelke, N. ., & Parati, N. . (2023). Advancements in Computing: Emerging Trends in Computational Science with Next-Generation Computing. International Journal of Intelligent Systems and Applications in Engineering, 12(7s), 546–559

[9] N. Rosmawarni, T. T, I. Ahmad, E. Ardhianto, D. Handayani and W. P. Sari, "Hyperparameter Tuning On Machine Learning Transformers For Mood Classification In Indonesian Music," 2023 International Conference on Informatics, Multimedia, Cyber and Informations System (ICIMCIS), Jakarta Selatan, Indonesia, 2023, pp. 639-643, doi: 10.1109/ICIMCIS60089.2023.10349008

[10] R. Akella and T. -S. Moh, "Mood Classification with Lyrics and ConvNets," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 2019, pp. 511-514, doi: 10.1109/ICMLA.2019.00095.

[11] N. Rosmawarni, T. T, I. Ahmad, M. Ardiansyah, I. Kumalasari and J. Suwarno, "Classic Machine Learning For Mood Classification In Indonesian Music," 2023 International Conference on Informatics, Multimedia, Cyber and Informations System (ICIMCIS), Jakarta Selatan, Indonesia, 2023, pp. 670-673, doi: 10.1109/ICIMCIS60089.2023.10349079.

[12] J. A. Ridoean, R. Sarno, D. Sunaryo and D. R. Wijaya, "Music mood classification using audio power and audio harmonicity based on MPEG-7 audio features and Support Vector Machine," 2017 3rd International Conference on Science in Information Technology (ICSITech), Bandung, Indonesia, 2017, pp. 72-76, doi: 10.1109/ICSITech.2017.8257088.

[13] K. Agnihotri, P. Chilbule, S. Prashant, P. Jain and P. Khobragade, "Generating Image Description Using Machine Learning Algorithms," 2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP), Nagpur, India, 2023, pp. 1-6, doi: 10.1109/ICETET-SIP58143.2023.10151472.

[14] D. Liu, "Music Mood Classification Algorithm Considering Simulated Annealing Algorithm," 2023 International Conference on Network, Multimedia and Information Technology (NMITCON), Bengaluru, India, 2023, pp. 1-5, doi: 10.1109/NMITCON58196.2023.10276104.

[15] T. -T. Dang and K. Shirai, "Machine Learning Approaches for Mood Classification of Songs toward Music Search Engine," 2009 International Conference on Knowledge and Systems Engineering, Hanoi, Vietnam, 2009, pp. 144-149, doi: 10.1109/KSE.2009.10.

[16] A. Ualibekova and P. Shamoi, "Music Emotion Recognition Using K-Nearest Neighbors Algorithm," 2022 International Conference on Smart Information Systems and Technologies (SIST), Nur-Sultan, Kazakhstan, 2022, pp. 1-6, doi: 10.1109/SIST54437.2022.9945814.

[17] Dhanashri Shete, Prashant Khobragade; An empirical analysis of different data visualization techniques from statistical perspective. *AIP Conf. Proc.* 29 September 2023; 2839 (1): 040017.

[18] M T Quasim, E H Alkhammash, M A Khan et al., "Emotion-based music recommendation and classification using machine learning with IoT Framework[J]", Soft Computing, vol. 25, no. 18, pp. 12249-12260, 2021.

[19] A Werner, "Organizing music organizing gender: algorithmic culture and Spotify recommendations[J]", Popular Communication, vol. 18, no. 1, pp. 78-90, 2020.

[20] M O'Dair and A Fry, "Beyond the black box in music streaming: the impact of recommendation systems upon artists [J]", Popular Communication, vol. 18, no. 1, pp. 65-77, 2020.

[21] A T Chodos, "What does music mean to Spotify? An essay on musical significance in the era of digital curation[J]", INSAM Journal of Contemporary Music Art and Technology, vol. 1, no. 2, pp. 36-64, 2019.

[22] A Garg, V Chaturvedi, A B Kaur et al., "Machine learning model for mapping of music mood and human emotion based on physiological signals[J]", Multimedia Tools and Applications, vol. 81, no. 4, pp. 5137-5177, 2022.

[23] E. P. Ijjina, "Classification of human actions using pose-based features and stacked auto encoder", Pattern Recognition Letters, vol. 83, pp. 268-277, 2016.

[24] V Moscato, A Picariello and G Sperli, "An emotional recommender system for music[J]", IEEE Intelligent Systems, vol. 36, no. 5, pp. 57-68, 2020.

[25] Afis Julianto, Andi Sunyoto and Ferry Wahyu Wibowo, "Optimasi Hyperparameter Convolutional Neural Network Untuk Klasifikasi Penyakit Tanaman Padi", Tek. Teknol. Inf. dan Multimed., vol. 3, no. 2, pp. 98-105, 2022.

[26] Y. A. Ali, E. M. Awwad, M. Al-Razgan and A. Maarouf, "Hyperparameter Search for Machine Learning Algorithms for Optimizing the Computational Complexity", Processes, vol. 11, no. 2, pp. 1-22, 2023.

[27] S. Zhang and N. Jiang, "Towards Hyperparameter-free Policy Selection for Offline Reinforcement Learning", Adv. Neural Inf. Process. Syst., vol. 16, no. NeurIPS, pp. 12864-12875, 2021.